DEM-based Illumination Simulation in a Web Map Service using Lunaserv

Caleb Hanger, Nick Estes, Mark Robinson, Ernest Bowman-Cisneros

Lunar Reconnaissance Orbiter Camera

Arizona State University

Introduction

The LROC SOC (Lunar Reconnaissance Orbiter Science Operations Center) developed its own implementation of WMS (Web Map Service), called Lunaserv [1].

Lunaserv features rendering engines for a wide variety of image layer types, including raster data, vector data (specified either via a plain text format developed specially for Lunaserv, or via Esri Shapefiles), and dynamically-generated, topography-based, synthetic illumination. The renderer responsible for the illumination layer is called map_illum_layer.

map_illum_layer was originally developed with the Moon in mind, but is usable for any planetary body, via the use of an appropriate DEM.

map_illum_layer needs several inputs, primarily 1) a DEM (Digital Elevation Map) in ISIS Cube format, whose data may be specified as either elevations or radius values, 2) a sub-solar point defined by longitude and latitude in degrees (if running via Lunaserv, a time can be passed in, and Lunaserv will derive the sub-solar coordinates using SPICE), and 3) the geographical region to be observed and synthesized. From these inputs and various pieces of optional information, map_illum_layer outputs an image in PNG format representing the synthesized illumination across the desired region.

map_illum_layer performs both shadowing (detecting which areas are completely obscured from sunlight by blocking terrain) and shading. The application also includes support for the user to choose between several photometric functions, and a straightforward mechanism for programmers to implement new photometric functions. Currently available are an implementation of the basic Lambertian law [2] and an implementation of the Lommel-Seeliger law [3].

How does it work?

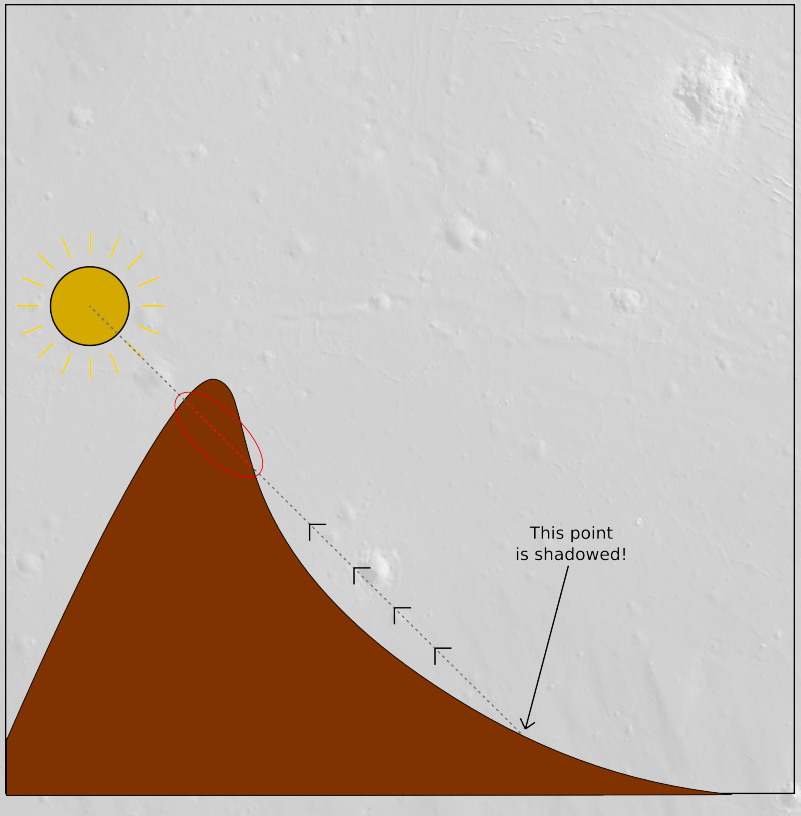
At the outermost level, map_illum_layer works by iterating through every pixel of desired output and deciding on a pixel-by-pixel basis how to color the pixel. For example, if the user requests a 2,000x2,000 image, map_illum_layer will execute 4,000,000 iterations through the main loop, one to decide how to color each of the 2,000 * 2,000 = 4,000,000 pixels.

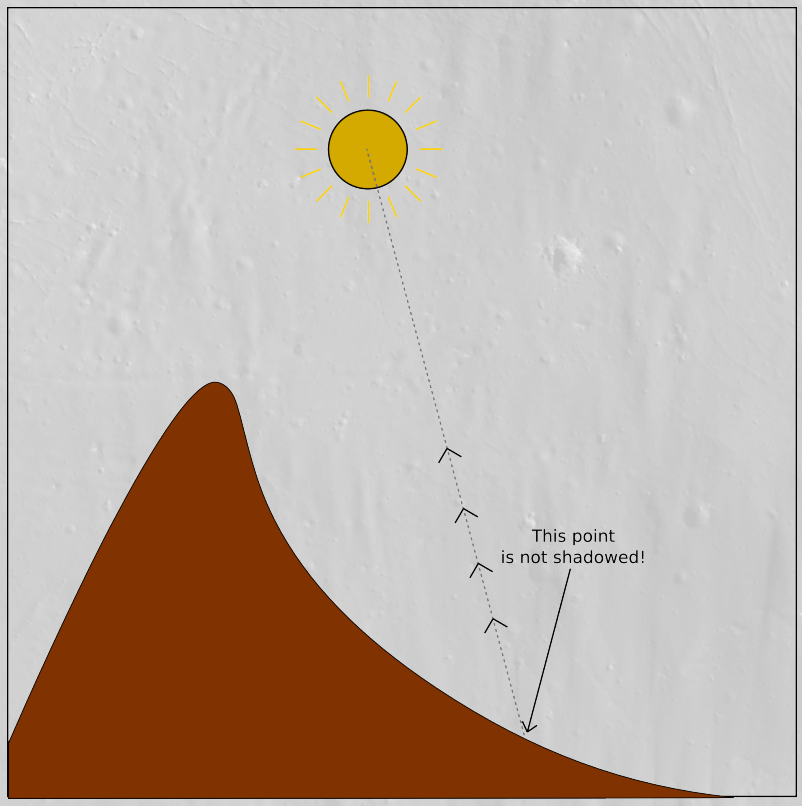
For each pixel, the first step is to convert the location from image space (pixel coordinates) to geographic space. map_illum_layer, like all of Lunaserv's renderers, uses the PROJ.4 library [4] to perform projection transformation. Thus, a wide variety of projections are supported, and the user of the application can choose an output projection using a PROJ.4 projection string (the default is equirectangular).

Once the pixel has been resolved to a geographic location, the next step is to check whether or not that particular point on the surface is shadowed. map_illum_layer uses a basic ray-tracing algorithm [5]. First, the longitude and latitude of the point are converted to a 3-element vector of rectangular coordinates; the DEM is also consulted in order to refine the vector based on the elevation of the surface point. Then, map_illum_layer steps from this surface point directly towards the location of the Sun in Cartesian space, checking points along the way to see whether or not they are below the surface -- if it finds at least one such point, then the original point is deemed shadowed, as a higher elevation terrain point has been found which blocks the original point. If map_illum_layer reaches a point where it can be safely assumed that continuing the search would be pointless (as we are getting too far away from the surface in our search), then it assumes that the point in question is not shadowed.

Examples of shadow determination

Note: The distance interval between checked points along the vector towards the Sun must be carefully chosen. Too coarse a search, and the algorithm might step past a small facet of Sun-blocking terrain in between checks. However, the finer the search, the longer it takes map_illum_layer to run; shadow-checking is where map_illum_layer usually spends a majority of its runtime. By default, map_illum_layer currently uses an interval of five times the pixel resolution of the input DEM, but this value can be overridden by the user.





How does it work? (cont'd)

If a pixel is determined to be shadowed, map_illum_layer writes a shadow DN (by default black, but overridable by the user) and continues to the next pixel. However, if the pixel is not shadowed, then map_illum_layer proceeds to compute three illumination-related angles -- incidence, emission, and phase -- which are required in order to execute a photometric function and thus determine shading.

map_illum_layer computes both incidence and emission angles with respect to local topography, in order to yield accurate results. The phase angle is then calculated as the angle between the vector toward the Sun and the vector toward the viewer.

In order to do this, map_illum_layer first locates the point of interest in the input DEM. Three pixels are chosen from the DEM -- a "center-pixel", a "left-pixel", and an "above-pixel". If the pixel of interest is not on the very top or left edge of the DEM, it is used as the center-pixel. If it does lie on one or both of these edges, an adjacent pixel (to the right, below, or both) is chosen as the center-pixel. The pixel above the center-pixel is the above-pixel, and the pixel to the left of the center-pixel is the left-pixel. The longitudes, latitudes, and elevations of these three pixels are fetched from the DEM, and with this information, map_illum_layer converts the three points into three-dimensional Cartesian coordinates that account for elevational difference. From here, it derives a vector from the center-pixel to the above-pixel, and another from the center-pixel to the left-pixel, then takes the cross product of the two vectors (in the same order) to obtain a surface normal vector that is topographically appropriate for the surface point in question. From this surface normal, an emission angle and an incidence angle which are both topographically accurate are derived.

How does it work? (cont'd)

Once the illumination angles have been computed, map_illum_layer proceeds to execute the photometric function that was selected by the user. By default, a simple Lambertian model is used, wherein the pixel is shaded based on the cosine of the incidence angle.

A Lommel-Seeliger-based model is also available; when using this option, map_illum_layer's behavior is to first cap the incidence and emission angles to a maximum of 88 degrees, in order to avoid wildly varying results that the formula yields when incidence and/or emission is too close to 90 degrees; without using this restraint, output values can range into the hundreds of thousands, but applying the restraint generally keeps output values between 0 and 1. Then, the expression:

$$\frac{cos(incidence)}{cos(emission) + cos(incidence)}$$

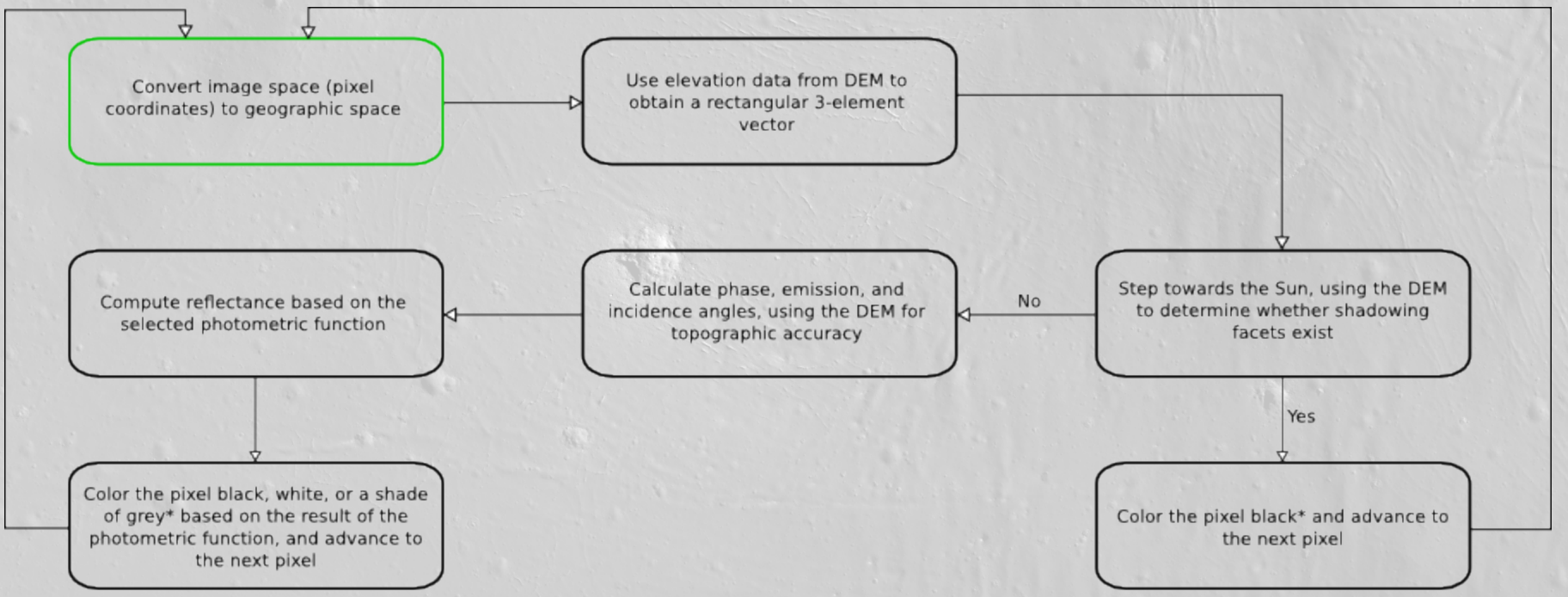
is computed. The result is constrained to the interval [0, 1] (if it is not already in this interval), and then passed through a "brightening" polynomial function:

$$f(x) = -x^2 + 2x$$

This function is the simplest polynomial which maps 0 to 0, 1 to 1, and 0.5 to 0.75, resulting in an overall brightening effect for the output.

After computing the result of the photometric function, the pixel is colored appropriately; by default, a result of 0 leads to a black pixel and a result of 1 leads to a white pixel. Everything in between is a shade of grey, scaled linearly based on the result from the photometric function. Both the maximum illumination and minimum illumination can be overridden by the user, and values in between are automatically computed by map_illum_layer using RGB values interpolated from the minimum and maximum.

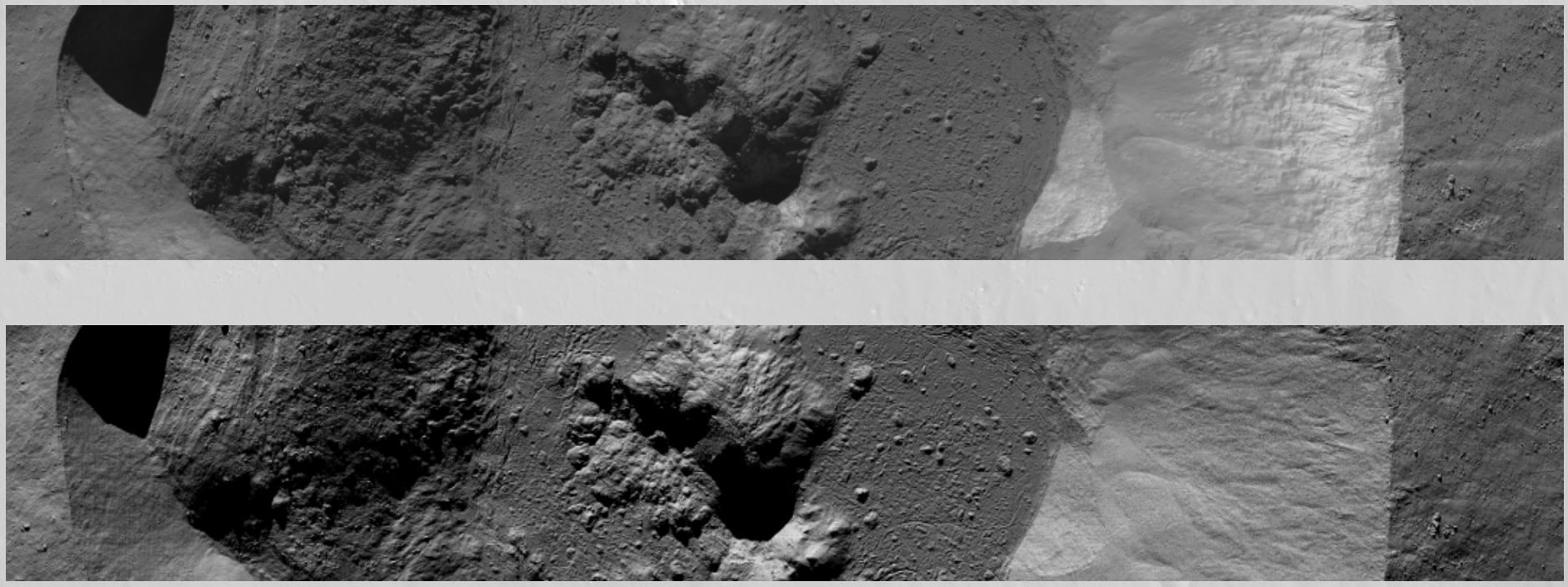
Overview of per-pixel logic



* Shadow color and dark/light shading colors are customizable by the user -- black for shadowing, and a black-white scale for shading, are defaults.

map_illum_layer example output

Following is a comparison between actual NAC (Narrow Angle Camera) imagery of Moore F crater at longitude 175°W, latitude 37.3°N (top) and an illumination map produced by map_illum_layer given the same sub-solar coordinates (bottom). The contrast is generally higher in the illumination map than in the actual image because map_illum_layer by default does not account for surface albedo. The illumination map was generated at an output resolution of 1000x6500 pixels or 6.5 megapixels, with a 2m/px DEM as input. The process took about four minutes and eight seconds on a current model workstation.



Uses

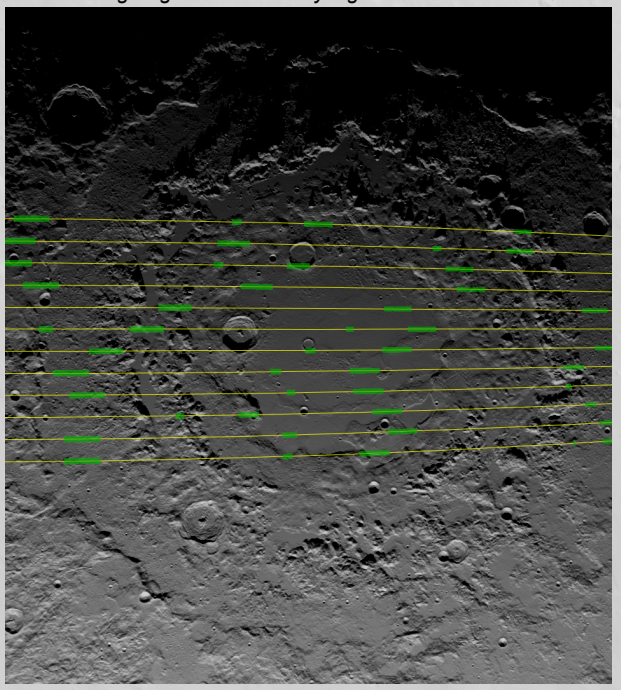
The illumination synthesis provided by map_illum_layer is used for a variety of benefits at the LROC SOC:

Camera operators can use the application to predict lighting conditions in upcoming days in LRO's flight path; this is extremely helpful in selecting worthwhile imaging targets.

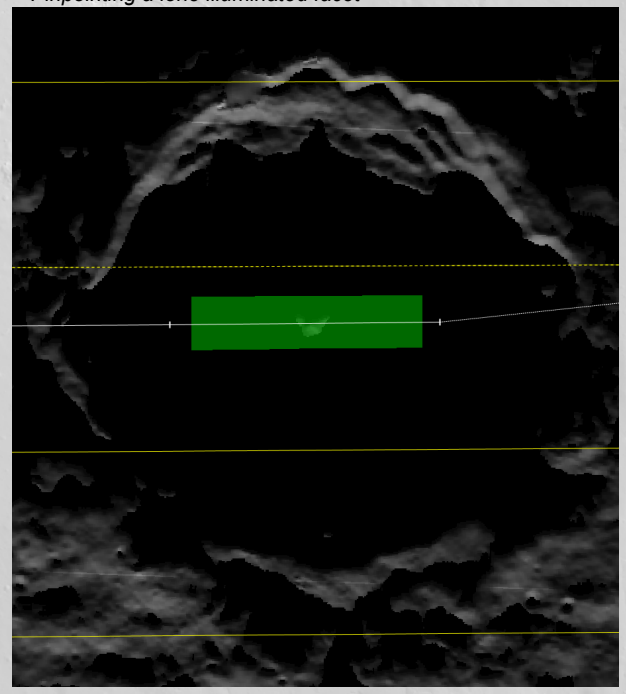
In particular, map_illum_layer is especially helpful in scheduling polar targeting campaigns.

Since map_illum_layer will accept any sub-solar point, including physically impossible locations (i.e. any point with a latitude significantly far from zero), research regarding planetary topography is furthered by the ability to synthesize a scene with artificial lighting conditions.

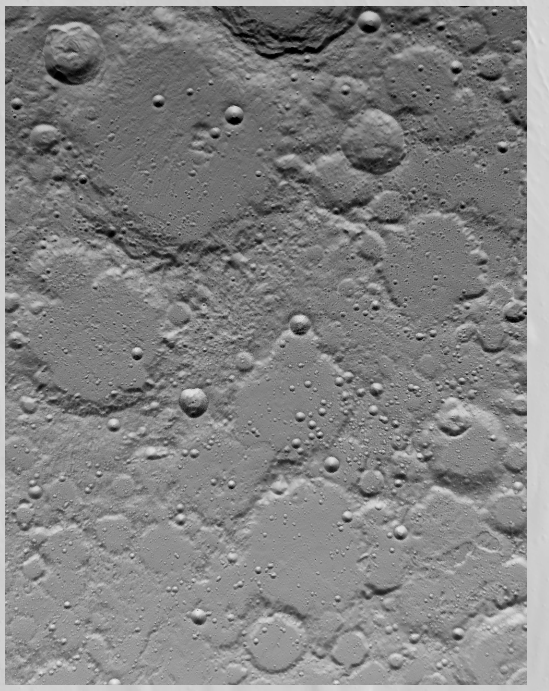
Scheduling targets near the day/night terminator



Pinpointing a lone illuminated facet



Illumination map centered on the north pole and simulating a sub-solar point of 0°E, 30°N.



Other features

map_illum_layer has undergone significant improvements in the short time since the writing of the abstract for this presentation. Particularly of note, the Lommel-Seeliger lighting model was not available until recently, and the method of surface normal computation (for determining local incidence angles) was completely re-designed for accurate results and to remove a potentially-confusing mathematical parameter.

Another feature recently added to map_illum_layer is the ability to specify an ISIS cube containing albedo values. These values have no effect on shadowing, but when shading is being performed, the original shading result is multiplied by the albedo value at the point in question to arrive at a final shading result.

map_illum_layer has the option to employ a simple but effective technique for reducing runtime when the input DEM is very large. On a system with ample RAM, the entire cube file can be pre-cached into a buffer in memory. The disk I/O cost of this initial pre-load is far outweighed by the time saved later, during processing.

Although map_illum_layer considers the Sun to be a point source and not a disc, it has been recently modified to consider the *edge* point of the Sun closest to the zenith, rather than the center. This provides increased accuracy, particularly in situations where the Sun is near the horizon and the small angular difference between the center of the Sun and the "top" edge of the Sun can cause a very noticeable difference in illumination. Since this difference is dependent on the distance to the Sun, the user can specify this distance; the default is one astronomical unit, appropriate for both Earth and the Moon.

Future plans

The next goal is to implement a photometric function for enhanced detail and realistic synthesis of lunar lighting conditions, using the models that Dr. Bruce Hapke has developed [6].

Want to try it out?

map_illum_layer, along with the entire Lunaserv package, is open source and can be downloaded from the Lunaserv website [7]. map_illum_layer is written in C, with liberal use of C99-only syntax. Provided some library dependencies are available, it should easily compile on any POSIX-compliant system with a recent version of the GNU C Compiler installed (other compilers supporting C99 will probably work as well).

map_illum_layer is also demonstrated on the web interface [8] to Lunaserv. If you are viewing the Moon or Mars (accessible via the "Object to view" control in the "Map Options" dialog box), you can enable the use of map_illum_layer to plot illumination data as follows. In the "Layers" dialog box, under the menu "Miscellaneous", enable the "GLD100 DEM based illumination" layer (for the Moon) or the "MOLA DEM based illumination" layer (for Mars).

References

[1] http://adsabs.harvard.edu/abs/2013LPICo1719.2609E

[2] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Lambert_s_cosine_law.html

[3] http://astrowww.phys.uvic.ca/~tatum/plphot/plphot01.pdf

[4] http://trac.osgeo.org/proj/

[5] http://www.codermind.com/articles/Raytracer-in-C++-Introduction-What-is-ray-tracing.html

[6] http://onlinelibrary.wiley.com/doi/10.1029/JZ068i015p04571/abstract

[7] http://lunaserv.lroc.asu.edu/

[8] https://webmap.lroc.asu.edu/